COP 3502 - 11/9/2023

Hash Tables

insert, search, delete } $O(\lg n)$ Balanced Binary Search Tree

Can we do better?

To use a hash table, need a hash function!

Properties of a hash function

① Input: Variable Length/Type (strings)

② Output: ~~Fixed~~ value in a fixed range
either $[0, \ldots n-1]$ for some int n
of a fixed # of bits. ← for crypto

It will be case there exist
different inputs x, x' s.t. $H(x) = H(x')$
↳ A COLLISION

Properties of Good Hash Functions

① Given 2 inputs x, x' the probability $H(x) = H(x')$ is close to $\frac{1}{n}$, n described above.

② If I give you y, it should be difficult for you to find any x s.t. $H(x) = y$.

Bad Hash function

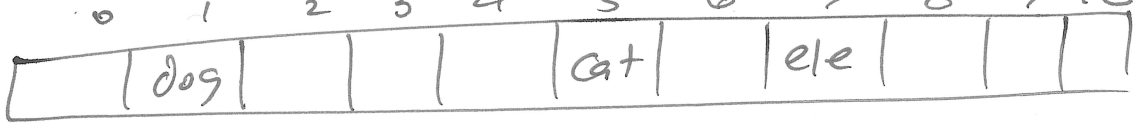
```
int f(char* word, int n) {
    int sum = 0;
    int len = strlen(word);
    for (int i = 0; i < len; i++)
        sum += (sum + word[i] - 'a') % n;
    return sum;
}
```

Reasonable Hash function

```
int f(char* word, int n) {
    int sum = 0, len = strlen(word);
    for (int i = 0; i < len; i++)
        sum = (26 * sum + word[i] - 'a') % n;
    return sum;
}
```

cat = $2 \times 26^2 + 0 \times 26^1 + 19 \times 26^0$

How do I use a hash function in a hash table?



insert(w) - 1. Calculate H(w). Put w in slot H(w)

search(w) - 1. Calculate H(w). Look in slot w.

f("cat") = 5

f("dog") = 1

f("sheep") = 2

f("elephant") = 7

f("tiger") = 1

ISSUE: Collisions!

How to deal w/ collisions

(1) Don't - overwrite old data

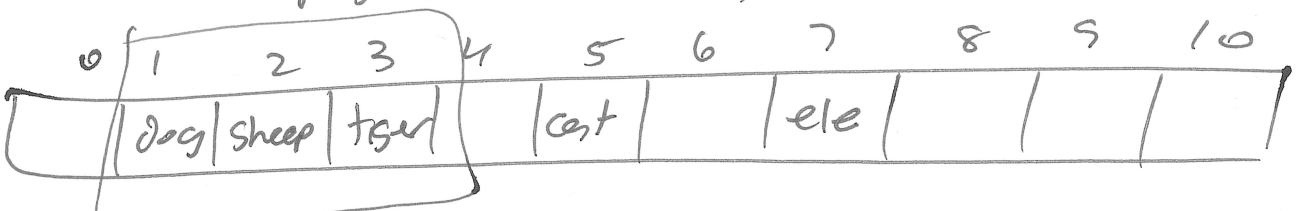
pro - easy to implement, fast

con - lossy

(2) Linear Probing - insert, search

if index i is full, then goto i+1, i+2, ... n-1, 0, 1, 2, ..

keep going until empty slot



f("tiger") = 1 → 1st open slot index 3

Search - must go until empty.

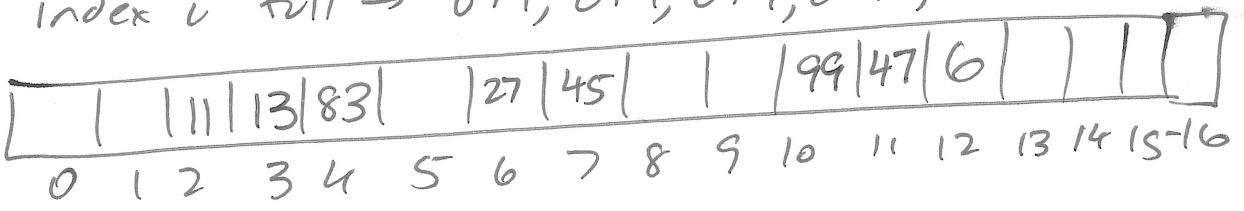
Run-times based on the "longest runs"

↓

CLUSTERS

(3) Quadratic Probing - insert, search (prime # table size)

index i full → i+1, i+4, i+9, i+16, ... %n



n=17

f(45) = 6

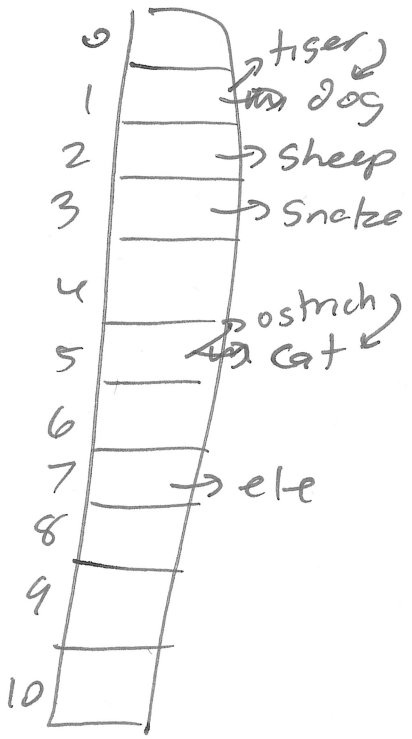
f(99) = 6

f(83) = 4

f(6) = 3

f(11) = 3 → (3+16) % 17 = 2

(4) Separate Chaining Hashing



f(tiger) = 1

each array slot is a pointer to a Linked List!!!

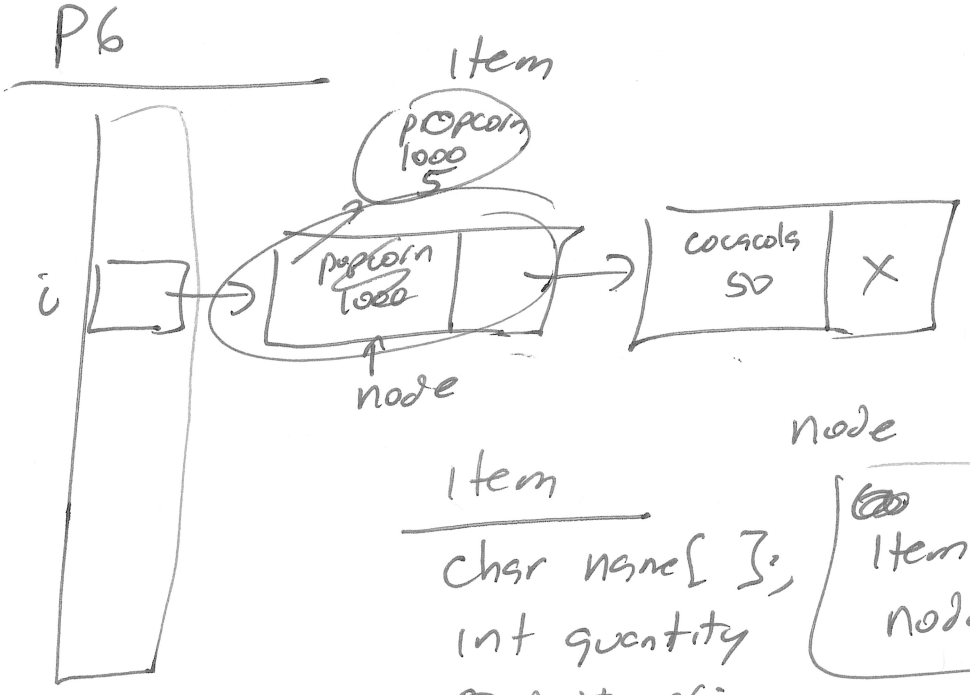
f("snake") = 3

f("ostrich") = 5

Run-time O(longest list) w/a good hash func

Longest List O(lg lg n) if we're storing n elements in a table size n.

P6



Item

char name[ ];
int quantity
~~float~~ int price

node

Item* itemptr
node* next;

n = odd prime

$i, i+1, i+4, i+9, \ldots i+\left(\frac{n-1}{2}\right)^2$

first $\frac{n+1}{2}$ locations we look

all of these are unique mod n.

Assume the opposite that $0 \le a < b \le \frac{n-1}{2}$

$$i + a^2 \equiv i + b^2 \pmod{n},$$

$$b^2 \equiv a^2 \pmod{n}$$

$$b^2 - a^2 \equiv 0 \pmod{n}$$

$$(b-a)(b+a) \equiv 0 \pmod{n}$$

n divides evenly into $(b-a)(b+a)$.

But if n is prime this means it either divides evenly into $(b-a)$ OR divides evenly into $(b+a)$.

$b-a > 0$ and $b-a \le \frac{n-1}{2}$

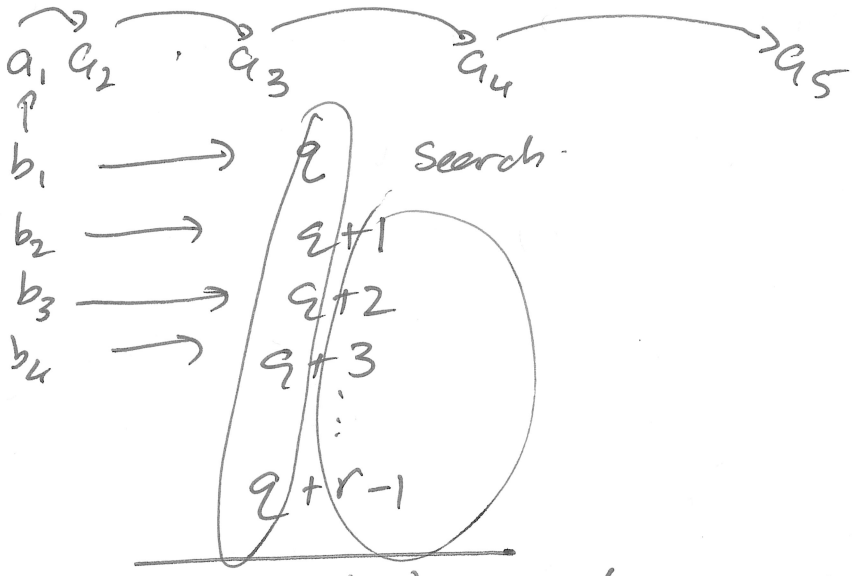
$b+a > 0$ and $b+a \le \frac{n-1}{2} + \frac{n-1}{2} = n-1$

CONTRADICTION

FND
EXAM
MAY
2022
SECTION B
QUESTION 2

q element tble

insert r new element



$$rq + \frac{(r-1)r}{2} = r\left(q + \frac{r-1}{2}\right)$$

$$O(r(q+r))$$

$$O(\max(rq, r^2))$$